**Java OOP Concepts**

Java is an object oriented programming language. Object oriented programming model visualizes everything as object. An object, in object oriented programming concept, is a real world entity which has both state and behavior. A class is a template for same type of objects. For example, A Car is a class which acts as a template for car1, car2, car3… objects. The whole object oriented programming paradigm or simply OOP model is based on four fundamental concepts. They are – Inheritance, Abstraction, Polymorphism and Encapsulation. These are often called as four main pillars of the object oriented programming model. Let’s see these OOP concepts one by one and how they are implemented in Java.

**1) Inheritance**

Inheritance, as name itself suggests, is used to inherit properties from parent class to child class. Using inheritance, you can reuse existing tried and tested code or else you can add more features to existing class without modifying it by extending it through its subclass.

In Java, inheritance is implemented by using extends keyword.

**class** SuperClass

**class** SubClass **extends** SuperClass

In the above program, SubClass is extending SuperClass. Hence, SubClass will have properties inherited from SuperClass along with its own properties. You can say that either we are reusing properties of SuperClass in SubClass or we are adding additional features to SuperClass by extending it through SubClass.

**2) Abstraction**

In computer science terms, abstraction means separating ideas from their actual implementations. Using abstraction, you define only ideas in one class so that those ideas can be implemented by its subclasses according to their requirements.

In Java, abstraction is implemented by abstract classes and interfaces.

**abstract** **class** AbstractClass

**class** SubClassOne **extends** AbstractClass

**class** SubClassTwo **extends** AbstractClass

In the above code, AbstractClass has only idea and that idea is implemented by SubClassOne and SubClassTwo according to their requirements.

You can also use interfaces in Java to implement abstraction.

**interface** Interface

**class** ClassOne **implements** Interface

**class** ClassTwo **implements** Interface

In the above code snippet, Interface has anIdea() which is implemented by ClassOne and ClassTwo according to their requirements.

Till Java 8, interfaces are allowed to have only abstract methods. From Java 8, they can also have concrete methods in the form of default and static methods thus reducing the gap between **abstract** classes and interfaces.

**3) Polymorphism**

Poly means many and morphs means forms. So, anything which has multiple forms is called as polymorphism. In computer science terms, any entity like operator or method or constructor which takes many forms and can be used for multiple tasks is called as polymorphism. For example, ‘+’ operator can be used for addition of two numbers as well as for concatenation of two strings.

In Java, there are two types of polymorphism – static polymorphism and dynamic polymorphism.

**Static Polymorphism :**

Any entity which shows polymorphism during compilation is called static polymorphism. In this type of polymorphism, object used is determined during compilation itself. Hence, it is also called as static binding or early binding.

Operator overloading, method overloading and constructor overloading are best examples of static polymorphism.

**Dynamic Polymorphism :**

Any entity which shows polymorphism at run time is called as dynamic polymorphism. It is also called as late binding or dynamic binding as object used is determined at run time. Method overriding shows dynamic polymorphism.

**4) Encapsulation**

Bundling of data and operations to be performed on that data into single unit is called as encapsulation. Encapsulation in Java can be achieved by including both variables (data) and methods (operations) which act upon those variables into a single unit called class.

Encapsulation is often used to hide important information from outside the world. It is called data hiding. This can be achieved by declaring all important variables as private and providing public getter and setter methods.